Prototyping Cloud

by Tracy Fullerton

Cloud is a student research project from the USC Game Innovation Lab with an unusual design goal: Create a tranquil, relaxing, and joyful emotional experience similar to the archetypal childhood daydream of flying in the sky and creating shapes in the clouds. As faculty advisor for the project, Tracy Fullerton worked with the team at each stage of the process to define and iterate the design. This award winning game has been downloaded more than 1.5 million times from www.thatcloudgame.com.

When we began working on Cloud, we had only an innovative design goal: to somehow evoke the feeling of relaxation and joy that you get when you lie back on the grass on a clear summer day and look up at the clouds, wandering across the sky. Everyone does this. And at some time or another, we've dreamed of flying up in the clouds and moving them, shaping them into funny creatures or smiley faces or lollipops, or whatever comes to mind. It seemed like entirely new territory for a game. It seemed risky and interesting. So we decided to give it a try.

But how to do it? The first step was to create a series of prototypes based on the core mechanic of flying and gathering clouds. These prototypes were implemented in using the Processing development environment and were iterated on over several generations, starting in 2D and moving into crude 3D to test control, camera, and gameplay integration.

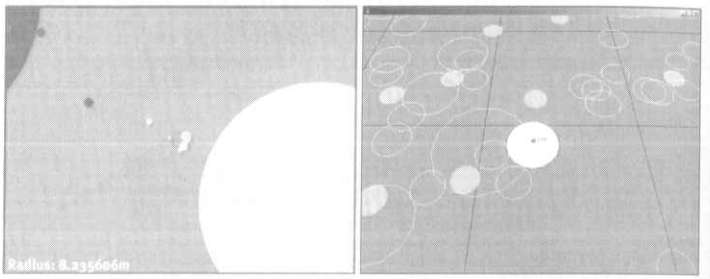

Figure 1 **2D gathering clouds prototype for Cloud, left, and control, camera, and rudimentary gameplay prototype, right**

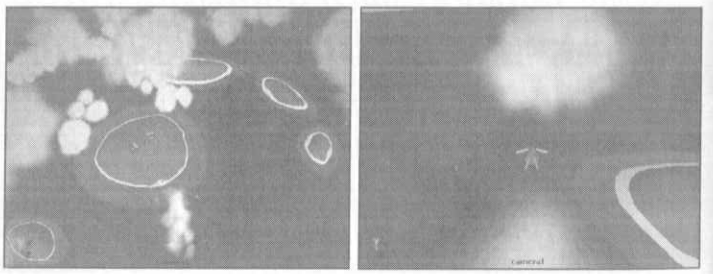

Figure 2 **Camera simulation prototype; left shows camera zoomed out for view of entire sky, right shows camera zoomed in to fly close to the character**

This core gameplay was tested by the team and a number of playtesters, and several conclusions were reached. The first was that the 2D perspective, while simple and practical, was not emotional enough. Although the final project had always been planned as a 3D game, there had been an open question of how to achieve a useable player viewpoint, and whether or not it made sense to use a 3D environment, but to lock the play within that environment to a two-dimensional plane within the 3D world.

At this point, the team began to sense that there was a conflict between

the desired clarity for gameplay (which called for a 2D playing field) and the equal desire for an emotional sense of freedom in flight (which called for freedom of movement within a 3D space). So we created a camera prototype in Maya that simply tested the idea of allowing the player to zoom the camera in and out at will. For example, we knew we wanted the player to be able to zoom far out to see what they had written in the sky, but we also wanted to be able to fly close up with the character, to feel the emotion of flight. As it turned out, this concept, especially when combined with another feature—"free flight" within the 3D space—solved both the practical interface issue and the emotional issue of flying close-up with the child.

In addition to experimenting with the core mechanic and viewpoint, at this point, the team began to envision a game without the traditional goals and conflict that drives most games. It would be a simple game that would encourage creativity and playfulness. To achieve this, we began designing the features that would allow players to draw and erase clouds in the sky as easy as chalk. Also, we began to realize that every aspect of the game needed to reinforce these positive emotions. The game needed to be relaxing and refreshing in its play as well as in its look and feel. So to eliminate all the psychic stress, there is no time pressure in the game, and failure is almost impossible. There are no elements that will trap players, and they can pick up and leave at any time with no repercussions.

While our gameplay prototyping was going on, focused on mechanics and camera controls, the programming team had several other hurdles they knew they had to face. The most important, obviously, was the simulation of believable, malleable, and computationally practical clouds. The team came up with an interesting solution: the use of a Leonard-Jones particle simulation underlying the clouds that would give them a dynamic underlying structure that would feel like playing with globs of mercury.

The first implementation of this concept was most useful in the fact that it proved we could create "clouds" out of clumps of dynamic particles—and that we would be able to support a lot of them. The images below of the particle simulation prototypes show the result of several thousand particles in a prototype environment that are (thankfully) not overtaxing the machine. These particles can be grabbed and shaped, much as the team had envisioned the cloud drawing feature.

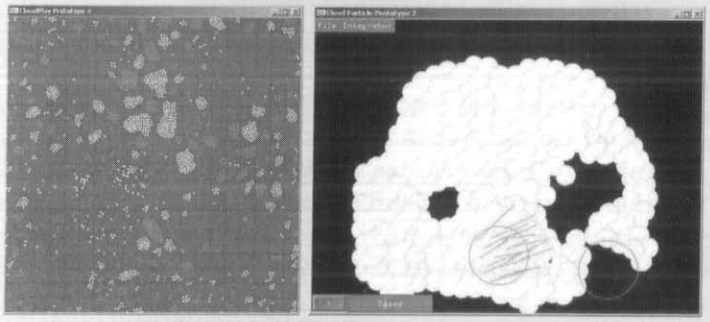

Figure 3 Particle simulations prototypes

The next stage of prototypes focused on making that underlying particle simulation feel more puffy. The previous figure shows such a test. In this version, tests revolved around using the clumps of clouds to draw faces, pictures, and an overall excitement about how the clouds would ultimately feel to play with started to permeate the team.

In addition to creating this underlying simulation, the team also implemented a billboarding method for rendering the cloud art onto the simulation. The following cloud simulation layer screenshots show

images from the final game with rendering turned on and off to demonstrate how the method mapped to the final simulation.

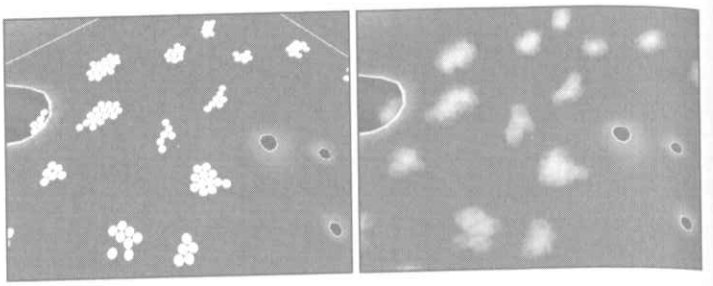

Figure 4 Cloud simulation layer (left) and with rendered clouds overlying simulation (right)

Throughout the prototyping process and into production, extensive playtesting, both by the team and by outside players, led to a number of changes and decisions. In the end, a number of technical features were cut to concentrate on the cloud simulation and the free flight controls. Concepts like wind, a day and night cycle, terrain features linked to cloud state, etc., were all prioritized under the response from playtesters for the need for a satisfyingly dynamic sky and intuitive flight controls.

These decisions are examples of the importance of playcentric design on the design and development process. While a traditional design team might have tried to implement all features, but with less depth in each, the iterative testing and reevaluation of the design based on overall experience made it clear that players were focused on the feel of the clouds and flight, not necessarily interactive terrain, day and night cycles, wind, or other missing elements.

In the end, Cloud proves that even a student research project can provide a strong model for gameplay innovation. Overall, though the design process had fits and starts throughout, and though we were not always certain of success, the methodology of playcentric design, and a clear design goal of finding new areas of emotional experience for games, brought this project safely to conclusion. So while risk was high, we had confidence in both the type of innovation we were exploring and the method by which we were doing our exploration.

As a designer, you need to make sure that you understand the capabilities of the controller for the platform you are designing to. This means creating a kinesthetic prototype and testing the controls until they are perfectly integrated with your gameplay. The article by designer Eric Zimmerman of Gamelab on page 330 describes how his team's desire to create an interesting new control idea scheme led to the idea for their game Loop. In this case, they created a digital prototype of the core mechanic—the "looping" control—and tested that thoroughly to make sure it was intuitive and fun before progressing further with the idea.

After you understand the input device, you need to think about how your game can best utilize it. You need to decide this in conjunction with your interface design, which we discuss on page 235. A good way to begin is to look at the list of procedures for your physical prototype. These procedures need to be translated into a set of digital controls. For example, in our first person shooter prototype, we had procedures for moving forward, backward, turning left, turning right, etc. We also had procedures for firing weapons, changing weapons, etc. Each of these will need to be mapped to a control. If you have a highly detailed set of controls, you will probably wind up grouping them under a menu system or other visual device that can be accessed using a single control or set of controls.

When you have decided how the controls will work, create a control table to make sure you have thought of everything. In one column, list the